CBCS SCHEME

USN | | | | | | | | | | |

BEE613B

Sixth Semester B.E./B.Tech. Degree Examination, June/July 2025
Embedded System Design

Time: 3 hrs. Max. Marks: 100

Note: 1. Answer any FIVE full questions, choosing ONE full question from each module.
2. M : Marks , L: Bloom's level , C: Course outcomes.

		Module – 1	M	L	C
Q.1	a.	Compare between general purpose Computing system and Embedded system.	06	L2	CO1
	b.	Explain the classification of Embedded systems based on Generation and Complexity and performance requirements.	08	L2,5	CO1
	c.	Give an outline on major application areas of Embedded system.	06	L2	CO1
		OR			
Q.2	a.	Explain with neat block diagram elements of Embedded system.	06	L2,5	CO1
	b.	Summarize the classification of program storage memory.	08	L2	CO1
	c.	Explain I2c Buss Interfacing with a neat sketch.	06	L2,5	CO1
		Module – 2			
Q.3	a.	Explain the characteristics of Embedded system.	10	L2,5	CO1
	b.	Discuss washing machine as an application specific Embedded system with functional block diagram.	10	L2,5	CO1
		OR			
Q.4	a.	Explain Automotive Communication Buses in Embedded system.	08	L2,5	CO1
	b.	Explain operational and non operational quality attributes of Embedded system.	12	L2,5	CO1
		Module – 3			
Q.5	a.	Explain the fundamental Issues in hardware software Co-Design.	10	L2,5	CO3
	b.	Explain 8 to 3 Encoder IC and Input / Output signal states with truth table.	05	L2,5	CO3
	c.	Summarize Analog and Mixed signal design in VLSI and Integrated circuit design.	05	L2	CO3
		OR			
Q.6	a.	Define flip-flops. Explain S – R flip-flop with truth table and logic circuit.	05	L1,2,5	CO3
	b.	Explain 8 to 1 multiplexer IC and Input / Output signal states with Truth table.	05	L2,5	CO3
	c.	List the different computational model in Embedded system. Explain the DFG model and CDFG model.	10	L1,2,5	CO3
		Module – 4			
Q.7	a.	What are the 2 basic approaches used for embedded firmware design. Explain super loop based Approach.	06	L1,2,5	CO4
	b.	Explain the types of files generated on cross - compilation.	08	L2,5	CO4
	c.	Explain the advantages of high level language based development.	06	L2,5	CO4

		OR			
Q.8	a.	Explain Monitor Program based firmware Debugging with neat sketch.	06	L2,5	CO4
	b.	Explain Assembly language to machine language conversion process with neat block diagram.	08	L2,5	CO4
	c.	Explain Disassembler /Decompiler in Embedded System.	06	L2,5	CO4
		Module – 5			
Q.9	a.	Explain operating system basics in Embedded system.	08	L2,5	CO2,5
	b.	Explain process strates and state transition in Embedded system with block diagram.	06	L2,5	CO2,5
	c.	Explain types of Multitasking in Embedded system.	06	L2,5	CO2,5
		OR			
Q.10	a.	Elaborate Pre-emptive scheduling in Embedded system.	08	L2,5	CO2,5
	b.	Explain the types of operating systems in embedded system.	06	L2,5	CO2,5
	c.	Explain Round Robin scheduling with block diagram.	06	L2,5	CO2,5

* * * * *

Module-1

Q.1) a) Compare between general purpose computing system and embedded system. (6 M)

	General purpose computing s/m	Embedded system
1.	It's a combination of generic h/w and a general purpose OS for executing variety of applications	It's a combination of special purpose h/w and embedded OS for executing a specific set of applications.
2.	Contains a GPOS	May or may not contain an OS for functioning
3.	Applications are programmable by the user	The firmware of the ES is preprogrammed and is not alterable by end user.
4.	Performance is the key deciding factor in the selection of the s/m. "Faster is Better"	Application specific requirements are the key deciding factors.
5.	Not built towards reduced operating power requirements, options for different levels of power management.	Highly tailored to take advantage of the power saving modes supported by the h/w and the OS.
6.	Response requirements are not time-critical	Response requirements are time critical.

b) Explain the classification of embedded systems based on Generation and complexity & performance requirements (8 M)

Soln. Classification of embedded s/ms based on Generation –

i) First Generation –

The early embedded s/m's were built around 8-bit MP's like 8085, and Z80 and 4-bit MC's. Simple in h/w ckts with firmware developed in assembly code.

Ex- digital telephone keypads, stepper M/r control

ii) Second Generation –

ES's are built around 16-bit MP's and 8 or 16 bit MC's. The instruction set were much more complex and powerful than the first generation. Some ES's contained embedded OS's for their operation.

Ex- DAS, SCADA s/ms

iii) Third Generation –

ES's are built around 32-bit MP's & 16-bit MC's. A new concept of application & domain specific processors / controllers like DSP's & ASIC's came in to design. The instruction set of processors became more complex & powerful and the concept of instruction pipelining also evolved. The processor market was flooded with different types of processors from different vendors. Processors like Intel Pentium

Motorola 68k etc. gained attention in high performance embedded requirements. Embedded s/m's spreads its ground to areas like robotics, media, industrial process control, networking etc.

iv) Fourth Generation -

The advent of SOC's, reconfigurable processors and multicore processors are bringing high performance, tight integration and miniaturization into the embedded device market. The SOC technique implements a total system on a chip by integrating different functionalities with a processor core on an integrated circuit.

c) Give an outline on major application areas of embedded system (6 M)

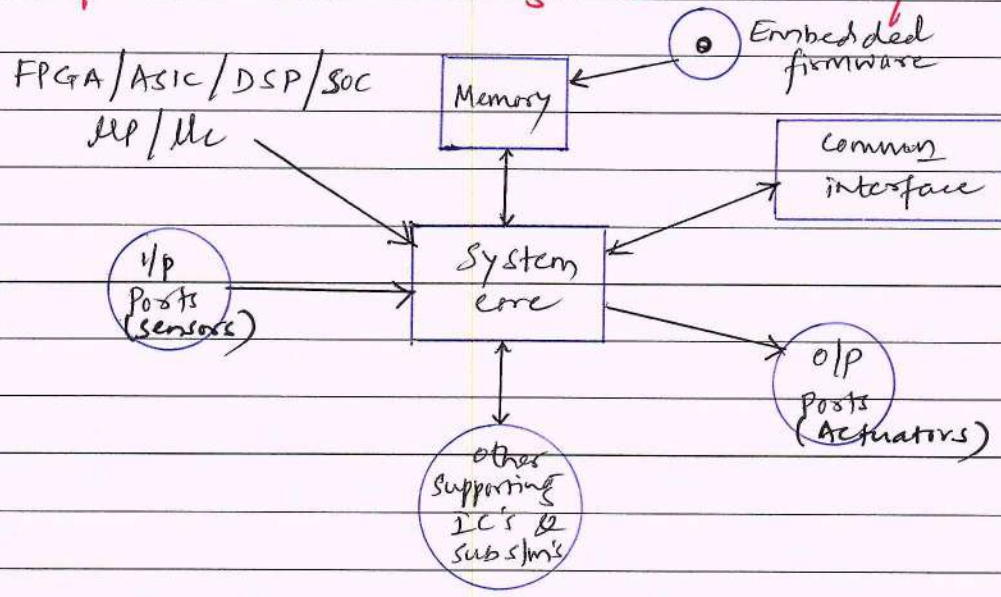
Soln: Major application areas of embedded system are listed below:

1) Consumer Electronics: camcorders, cameras, etc.
2) Household appliances: TV, DVD players, washing machine, refrigerator, Mwave oven
3) Home automation & security s/m's: AC, sprinklers, intruder detection, alarms, CCTV,
4) Automotive industry: ABS, Engine control, ignition s/m's, automatic navigation s/m's
5) Telecom: Cellular telephones, handset multimedia applications, etc.
6) Computer peripherals: Printers, scanners, fax m/c's
7) Computer n/w'ing s/m's: n/w routers, switches, hubs, etc

8) Health care : Scanners, EEG, ECG machines etc
9) Measurement & Instrumentation : Digital multimeters, digital CRO's, PLC s/m's
10) Banking & retail : ATM, POS machines
11) Card readers : Barcode, smart card readers, etc
12) Wearable devices : smart watch, smart screens etc
13) Cloud computing and IoT

Q.2) a) Explain with neat diagram elements of ES (6M)

Soln.



A typical embedded s/m contains a single chip controller, which acts as the master brain of the s/m. The controller can be a MP, or a MC or a FPGA device or a DSP or an ASIC.

Embedded hardware/software systems are basically designed to regulate a physical variable or to manipulate the state of some devices by sending some control signals to the actuators or devices connected to the o/p ports of the s/m, in response to the i/p signals provided by the end users or sensors

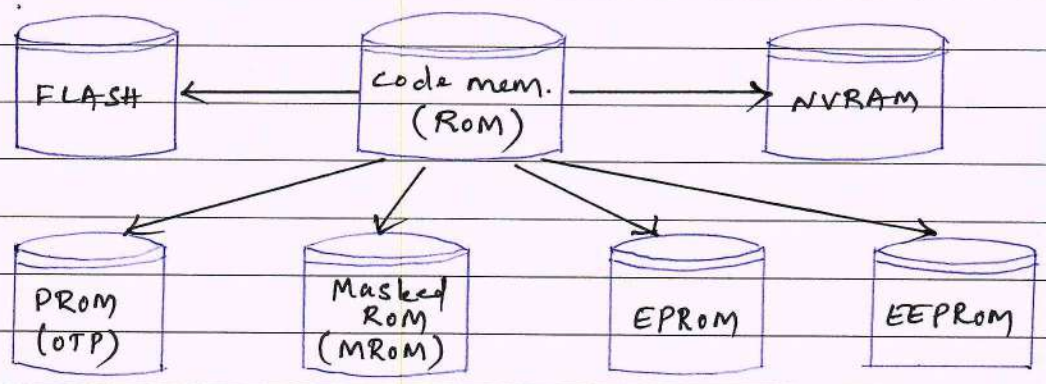
which are connected to the I/P ports. Hence an ES can be viewed as a reactive s/m. The control is achieved by processing the information coming from the sensors and user interfaces, and controlling some actuators that regulate the physical variable.

Key boards, push button switches etc. are examples for common user interface input devices whereas LED's, LCD's, piezoelectric buzzers, etc, are examples for common user interface O/P devices for a typical embedded s/m. It should be noted that it is not necessary that all embedded systems should incarporate these I/O user interfaces. It solely depends on the type of the application for which the embedded s/m is designed.

The memory of the s/m is responsible for holding the algorithm & other important configucation details. The most common types of memories used in embedded s/m's for control algorithm storage are OTP, PROM, UVEPROM, EEPROM and FLASH. Depending on the control application, the memory size may vary from a few bytes to megabytes.

b) Summarize the classification of Storage memory (8M)

Soln.



The program memory or code storage memory of an ES stores the program instructions and it can be classified in to different types as per the block diagram shown.

i) FLASH - FLASH is the latest ROM technology & is the most popular ROM technology used in todays embedded designs. FLASH memory is a variation of EEPROM technology. It combines the reprogrammability of EEPROM and the high capacity of standard ROM's. FLASH memory is organized as sectors or pages. It stores information in an array of floating gate MOSFET's. The erasing of memory can be done at sector level or page level without affecting the other sectors or pages. Each sector/page should be erased before re-programming.

ii) NVRAM - Non. Volatile RAM is a random access memory with battery backup. It contains static RAM based memory and a minute battery for providing supply to the memory in the absence of external power supply. The memory & battery are packed together in a single package. The lifespan of NVRAM is expected to be around 10 years.

iii) PROM (OTP) - PROM (or OTP) is programmed by the end users but not by manufacturers. PROM has nichrome or polysilicon wires arranged in a matrix. These wires can be functionally viewed as fuses. It is programmed by a PROM programmer which selectively burns the fuses according to the bit pattern to be

stored. Fuses which are not blown / burned represents a logic '1' whereas fuses blown / burned represents a logic '0'. OTP is widely used for commercial production of embedded s/m's whose proto-typed versions are proven and the code is finalised. OTP's can not be reprogrammed.

iv) Masked ROM (MROM) – MROM is a one-time programmable device. MROM makes use of the hardwired technology for storing data. The device is factory programmed by masking and metallisation process at the time of production itself, according to the data provided by the end user. The primary advantage of this is low cost for high volume production. They are the least expensive type of solid state memory. MROM is a good choice for storing embedded firmware for low cost embedded devices. MROM is permanent in bit storage, it is not possible ~~for~~ to alter the bit information.

v) EPROM – EPROM gives the flexibility to re-program the same chip. It stores the bit information by charging the floating gate of an FET. Bit information is stored by using an EPROM programmer, which applies high vg. to charge the floating gate. EPROM contains a quartz crystal window for erasing the stored information. If the window is exposed to UV rays for a fixed duration, the entire memory will be erased.

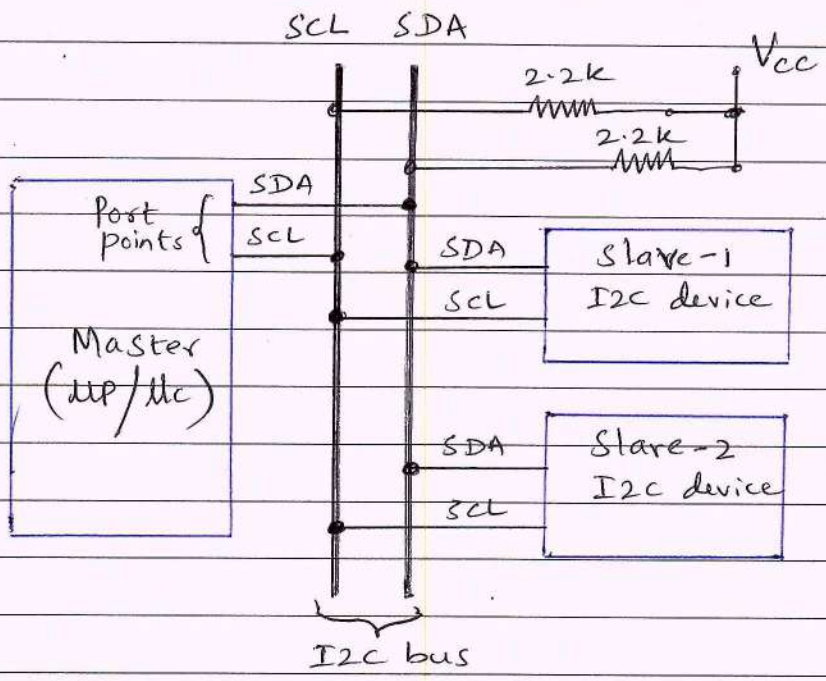
vi) EEPROM - As the name indicates, the information contained in EEPROM can be altered by using electrical signals at the register / Byte level. They can be erased and reprogrammed in-circuit. These chips include a chip erase mode and in this mode they can be erased in a few milliseconds. It provides greater flexibility for s/m design. The only limitation is their capacity is limited when compared with the standard ROM.

c) Explain I2C Bus interfacing with a neat sketch (6M)

Soln.

The Inter Integrated Circuit (I2C) bus is a Synchronous bidirectional, half duplex, two wire serial interface bus. The I2C bus comprises of two bus lines, namely; Serial Clock - SCL and Serial Data - SDA. SCL line is responsible for generating synchronization clock pulses and SDA is responsible for transmitting the serial data across devices. I2C is a shared bus s/m to which many number of I2C devices can be connected. Devices connected to the I2C bus can act as either "Master" device or "Slave" device. The master device is responsible for controlling the comming by initiating / terminating data transfer, sending data and generating necessary synchronization clock pulses. "Slave" devices wait for the commands from the master and respond upon receiving the commands. Master & Slave devices can act as either tx or Rx.

I2C supports multi masters on the same bus.



The I2C bus built around an i/p buffer and an open drain or collector transistor. When the bus is in the idle state, the open drain / collector tr. will be in the floating state and the o/p lines (SDA & SCL) switch to the "high impedance" state. For proper operation of the bus, both the bus lines should be pulled to the supply vg. using pull-up resistors. With pull-up resistors, the o/p lines of the bus in the idle state will be high.

The address of a I2C device is assigned by hard wiring the address lines of the device to the desired logic level. The address to various I2C devices in an embedded device is assigned and hardwired at the time of designing the embedded hardware.

Module-2

Q3) a) Explain the characteristics of embedded system. (10M)

Soln.

Embedded s/m possess certain specific characteristics and these characteristics are unique to each embedded s/m.

The characteristics of an embedded s/m are;

i) Application & domain specific -

Each embedded s/m is designed to perform a set of defined fns and they are developed in such a manner to do the intended fns only. They can not be used for any other purpose. It is the major criterion which distinguishes an embedded s/m from a general purpose computing s/m.

ii) Reactive and Real time -

Embedded s/m's are in constant interaction with the real world thro' sensors and user defined input devices. Any event in the real world are captured by the sensors or i/p devices in real time & the control algorithm running inside the unit reacts in a designed manner to bring the controlled o/p variables to the desired level. The event may be a periodic one or an unpredicted one. Embedded s/m's produce changes in o/p in response to the changes in the i/p. So they are generally referred as reactive systems.

Real time s/m operation means the timing behaviour of the s/m should be deterministic. A real time s/m should not miss any deadlines

for tasks or operations. Embedded applications or s/m's which are mission critical, like flight control, s/m's, ABS etc. are examples of real time s/m's. The design of an embedded real time s/m should take the worst case scenario in to consideration.

iii) Operates in Harsh environment -

The environment in which the embedded s/m deployed may be a dusty one or a high temp. zone or an area subject to vibrations or shocks. ES's placed in such areas should be capable to withstand all these adverse operating conditi-ons. The design should take care of the operating conditions of the area where the s/m is going to implement. Power supply fluctuations, corrosion and component aging, etc. are the other factors that need to be taken in to consideration for embedded s/m's to work in harsh environments.

iv) Distributed -

The term distributed means that the embedded s/m's may be part of larger s/m's. Many numbers of such distributed embedded s/m's form a single large embedded control unit. Ex- an automatic vending machine. The vending m/c contains a card reader, a vending unit, display unit etc. Each of them are independent units but they work together to perform the overall vending fn.

v) Small size & weight -

Product aesthetics is an important factor in

choosing a product. Ex- While buying a new phone, a comparative study on the pro's & con's of the products may be done by the buyer. Definitely the product aesthetics (size, weight, shape, style, etc) will be one of the deciding factors to choose a product.

vi) Power Concerns -

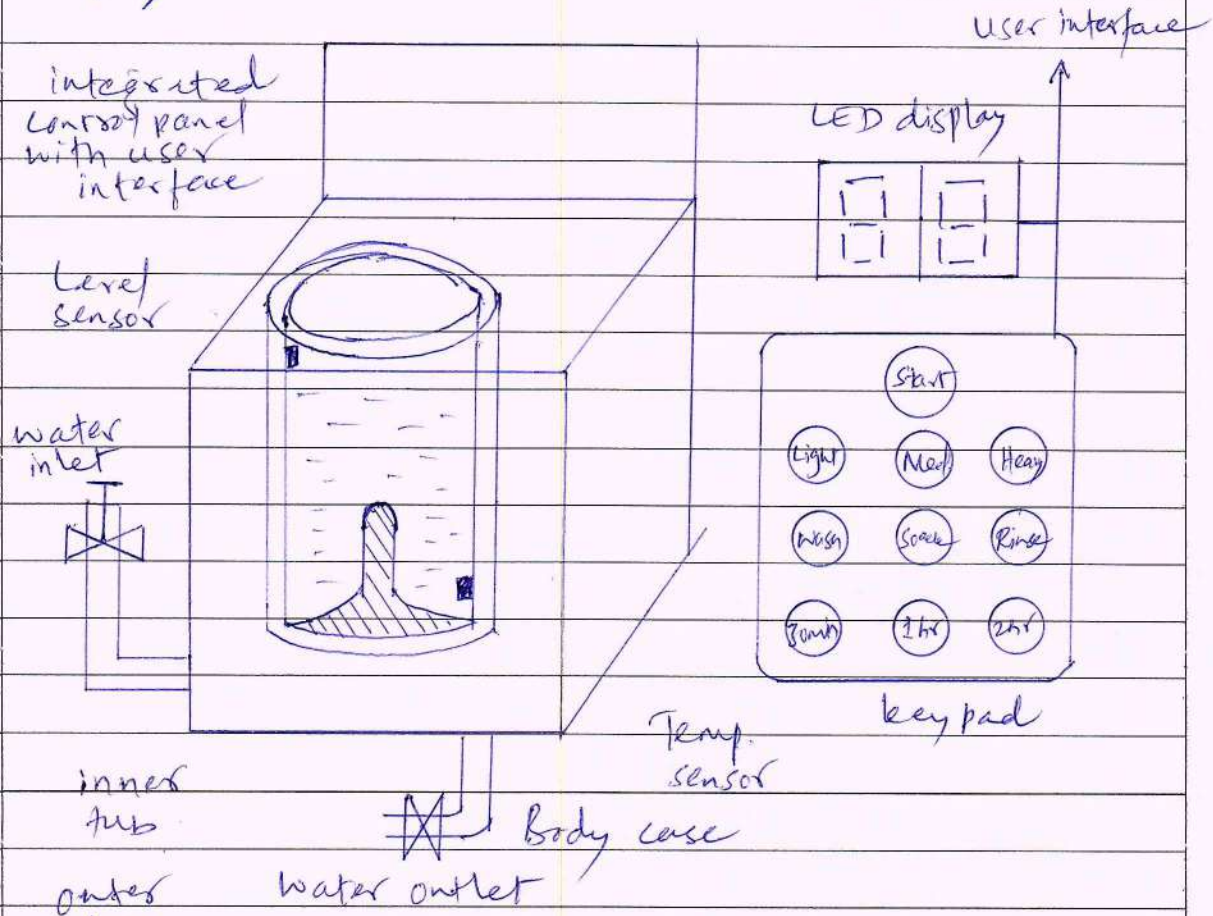
Power management is another important factor that needs to be considered in designing ES's. ES's should be designed in such a way as to minimize the heat dissipation by the s/m. The production of high amount of heat demands cooling requirements like cooling fans which in turn occupies additional space and make the s/m bulky. Now a days ultra low power components are available in the market. Select the design according to the low power components like low dropout regulators, and controllers / processors with power saving modes. Also power management is a critical constraint in battery operated application. The more the power consumption, the less is the battery life.

b) Discuss washing machine as an application specific Embedded s/m with functional block diagram (10 M)

Soln.

Washing machine as an application specific ES consists of sensors, actuators, control unit and application specific user interfaces like keyboards,

display units etc.



The actuator part of the washing m/c consists of a motorised agitator, tumble tub, water drawing pump and inlet valve to control the flow of water in to the unit. The sensor part consists of water temp. sensor, level sensor, etc. The control part contains a MP / MC based board with interfaces to the sensors & actuators. The sensor data is fed back to the control unit and the control unit generates the necessary actuator o/p's. The control unit also provides connectivity to user interfaces like keypad for setting the washing time, selecting the type of material to be washed

like light, medium, heavy duty, etc. User feedback is reflected thro' the display unit and LED's connected to the control board.

The integrated control panel consists of a µp/µc based board with I/O interfaces and a control algorithm running in it. Input interface includes the keyboard which consists of wash type selector viz; Wash, Spin & rinse, Cloth type selector and washing time setting etc. The O/p interfaces consists of LED/LCD displays, status indication LEDs, etc. connected to the I/O bus of the controller. The other types of I/p interfaces which are invisible to the end user are different kinds of sensor interfaces, namely water temp. sensor, water level sensor, etc and actuator interface including motor control for agitator and tub movement control, inlet water flow control etc.

Q.4) a) Explain automotive communication buses in embedded system (8 M)

Soln

The different types of automotive communication buses in embedded system are;
CAN, LIN & MOST

i) Controller Area Network (CAN) -

The CAN bus was originally proposed by Robert Bosch. It supports medium speed & high speed data transfer. CAN is an event-driven protocol interface with support for error handling in data

transmission. It is generally employed in safety s/m like airbag control; power train s/m's like engine control, ABS; and navigation s/m's like GPS.

ii) **Local Interconnect Network (LIN) -**

LIN bus is a single master, multiple slave communication interface. LIN is a low speed, single wire comm^n interface with support for data rates upto 20 kbps and is used for sensor/actuator interfacing. LIN bus follows the master communication triggering technique to eliminate the possible bus arbitration problem that can occur by the simultaneous talking of different slave nodes connected to a single interface bus. LIN bus is employed in applications like mirror controls, fan controls, seat positioning controls, window controls, and position controls where response time is not a critical issue.

iii) **Media Oriented System Transport (MOST) bus -**

MOST is targeted for high bandwidth automotive multimedia n/w'ing (ex. audio/video, infotainment s/m interfacing), used primarily in European cars. A MOST bus is a fibre-optic over-head point-to-point n/w implemented in a star, ring or daisy-chained topology over optical fibre cables. The MOST bus specifications define the physical layer as well as the application layer, n/w layer and media access control. MOST bus is an optical fibre cable connected between EOC & OEC, which would translate in to the optical cable MOST bus.

b) Explain operational and non-operational quality attributes of embedded system (12 M).

Soln

The operational quality attributes represent the relevant quality attributes related to the embedded s/m when it is the operational mode or online mode, viz;
Response, Throughput, Reliability, Maintainability, Security & Safety.

i) Response - Response is a measure of quickness of the s/m. It gives an idea about how fast the system is tracking the changes in input variables. Most of the ES's demand fast response which should be almost real time. (Ex- flight control application). Any response delay in the s/m will create potential impact to the safety of the flight as well as the passangers.

ii) Throughput - Throughput deals with the efficiency of the s/m. In general, it can be defined as the rate of production of operation of a defined process over a stated period of time. The rates can be expressed in terms of units of products, batches produced, or any other meaningful measurements. Throughput is generally measured in terms of "Benchmarks". A "Benchmark" is a refrence point by which something can be measured. Benchmarks can be a set of performance criteria that a product is expected to meet or a standard product that can be used for comparing other products of the same product line.

iii) Reliability - Reliability is a measure of how much % you can rely upon the proper functioning of the s/m or what is the % susceptibility of the s/m to failures.

Mean Time Between Failures (MTBF) and Mean Time To Repair (MTTR) are the terms used in defining system reliability. MTBF gives the freq. of failures in hrs/weeks/months. MTTR specifies how long the s/m is allowed to be out of order following a failure. For an embedded s/m with critical application need, it should be of the order of minutes.

iv) Maintainability - Maintainability deals with support and maintenance to the end user or client in case of technical issues & product failures or on the basis of a routine s/m checkup. Reliability & maintainability are considered to be as two complementary disciplines. A more reliable s/m means a s/m with less corrective maintainability requirements and vice versa. As the reliability of the s/m ↑ces, the chances of failure and non-functioning also reduces, there by need for maintainability is also reduced. Maintainability is closely related to the system availability.

Maintainability can be broadly classified in to two categories, viz; Scheduled or periodic maintenance (preventive maintenance) and Maintenance to unexpected failures (corrective maintenance). Some embedded products may use consumable components or may contain components which

are subject to wear & tear and they should be replaced on a periodic basis.

v) Security - "Confidentiality", "Integrity" and "Availability" are the three major measures of information security. Confidentiality deals with the protection of data & application from unauthorized disclosure. Integrity deals with the protection of data and application from unauthorised modification. Availability deals with the protection of data and application from unauthorized ~~users.~~ modification. (users).

vi) Safety - "Safety" & "security" are two confusing terms. Sometimes both termed to be a single attribute. But they represent two unique aspects in quality attributes. Safety deals with the possible damages that can happen to the operators, public and the environment due to the breakdown of an embedded s/m or due to the emission of radioactive or hazardous materials from the embedded products. The breakdown of ES may occur due to a h/w failure or a firmware failure. Safety analysis is a must in product engineering to evaluate the anticipated damages and determine the best course of action to bring down the consequences of the damages to an acceptable level. Some of the safety threats are sudden and some of them are gradual.

Non-operational Quality attributes are; Testability & debug-ability, Evolvability, Portability, Time to prototype & market and per unit & total cost.

i) Testability & Debug-ability -

Testability is the degree to which an embedded s/m can be easily tested to verify that it works correctly. For an embedded product, testability is applicable to both the embedded h/w & firmware. Embedded h/w testing ensures that the peripherals and the total h/w fits in the desired manner, whereas firmware testing ensures that the firmware is functioning in the expected way.

Debug-ability is a means of debugging the product as such for figuring out the probable sources that create unexpected behaviour in the total s/m. Debug-ability has two aspects in the embedded s/m development context, namely, h/w level debugging & firmware level debugging. H/w debugging is used for figuring out the issues created by h/w problems whereas firmware debugging is employed to figure out the probable errors that appear as a result of flaws in the firmware.

ii) Evolvability -

Evolvability is a term which is closely related to biology. Evolvability is referred as the ~~non-volatile~~ non-heritable ~~violation~~ variation. For an embedded s/m, the quality attribute "Evolvability" refers to the ease with which the embedded product can

modified to take advantage of new firmware or hardware technologies.

iii) Portability - Portability is a measure of "independence". An embedded product is said to be portable if the product is capable of functioning "as such" in various environments, target processors / controllers and embedded OS. The ease with which an embedded product can be ported on to a new platform is a direct measure of the re-work required. A standard embedded product should always be flexible & portable. In embedded products, the term porting represents the migration of the embedded firmware written for one target processor to a different target processor. Re-compiling the programs for the new target processor generates the new target processor-specific ~~settings~~. Machine codes.

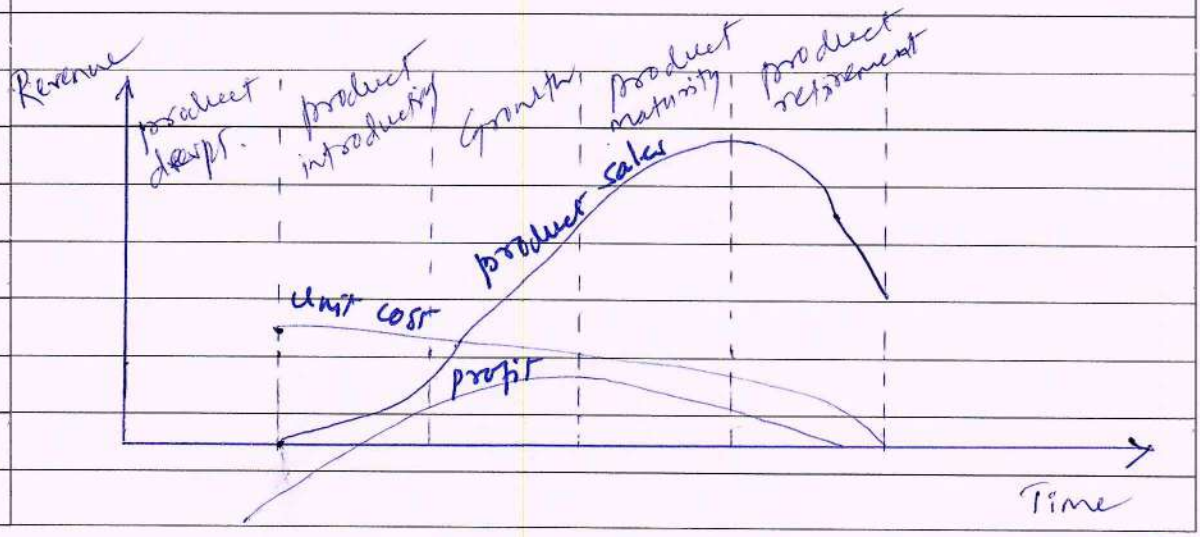
iv) Time-to-prototype and market -

Time-to-market is the time elapsed between the conceptualisation of a product and the time at which the product is ready for selling or use. The commercial embedded product market is highly competitive and time to market product is a critical factor in the success of a commercial embedded product. Embedded technology is one where rapid technology change is happening. Product prototyping helps a lot in reducing time-to-market. The time to prototype is also another critical factor. If the prototype is

developed faster, the actual estimated development time can be brought down significantly. In order to shorten the time to prototype, make use of all possible options like the use of off-the-shelf components, re-usable assets, etc.

v) Per unit cost & Revenue- cost is a factor which is closely monitored by both end user & product manufacturer. Cost is a highly sensitive factor for commercial products. Any failure to position the cost of a commercial product at a nominal rate, may lead to the failure of the product in the market. Proper market study and cost benefit analysis should be carried out before taking a decision on the per-unit cost of the embedded product. From a designer / product development company perspective, the ultimate aim of a product is to generate marginal profit. So the budget and total system cost should be properly balanced to provide a marginal profit.

vi) The Product Life Cycle (PLC) –



Every embedded product has a PLC which starts with the design & development phase. The product idea generation, prototyping, roadmap definition, actual product design & development are the activities carried out during this phase. During the design & development phase there is only investment & no returns. Once the product is ready to sell, it is introduced to the market. This is product induction stage. During the initial period the sales & revenue will be low. Product sales & revenue increases with time. During the maturity phase, the growth of sales will be steady and the revenue reaches at its peak. The product retirement / decline phase starts with the drop in sales volume, market share, revenue.

Module-3

Q.5) a) Explain the fundamental issues in software hardware co-design (10 M).

Soln:

Fundamental issues in hardware-software co-design are;

i) Selecting the model -

In h/w-s/w codesign models are used for capturing & describing the s/m characteristics. A model is a formal s/m consisting of objects and composition rules. It is hard to make a decision on which model should be followed in a particular design s/m. Most of the design engineers switch between a variety of models from the requirements specification to the implementation aspect of the s/m design.

ii) Selecting the architecture -

The architecture specifies how a s/m is going to implement in terms of number & types of different components and the interconnection among them. Controller architecture, data path architecture, CISC, RISC, VLIW, SIMD, MIMD etc. are the commonly used architectures in s/m design. Some of them fall in to ASIC, while others fall in to general purpose architecture class. or parallel processing class.

iii) **Selecting The language -**

A programming language captures a computational model & maps it in to architecture A model can be captured using multiple programming languages like C, C++, C#, Java etc for s/w implementation & Languages like VHDL, systems C, Verilog etc. for h/w implementations. C++ is a good candidate for capturing and object oriented model.

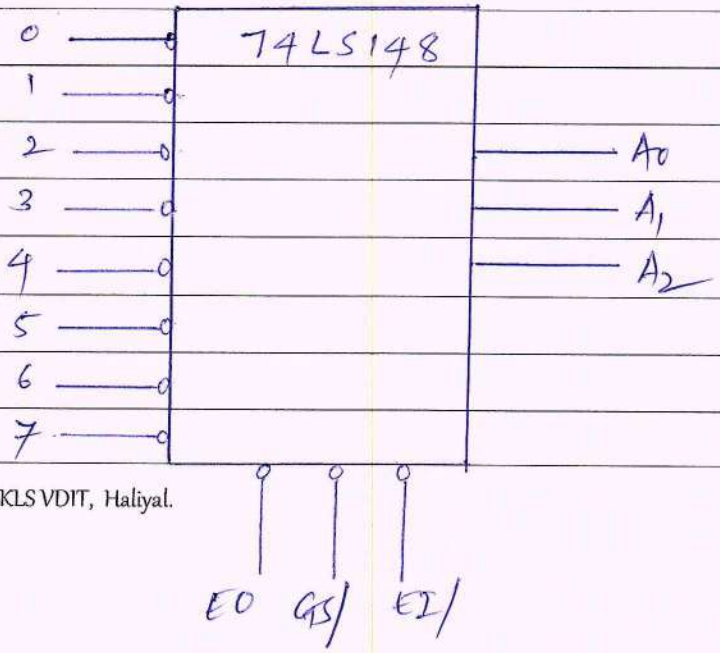
vi) **Partioning s/m requirements in to h/w & s/w -**

From an implementing perspective, it may be possible to implement the s/m requirements in either h/w ≃ s/w. It is a tough decision making task to figure out which one to opt. Various h/w & s/w trade-off are used for making a decision on the h/w - s/w partioning.

b) Explain 8 to 3 encoder IC and input/output signals states with truth table. (5 M)

Soln

8:3 Encoder IC -



Truth Table

Input									Output				
7	6	5	4	3	2	1	0	EI	GS	EO	A2	A1	A0
1	1	1	1	1	1	1	0	0	0	1	1	1	1
1	1	1	1	1	1	0	1	0	0	1	1	1	0
1	1	1	1	1	0	1	1	0	0	1	1	0	1
1	1	1	1	0	1	1	1	0	0	1	1	0	0
1	1	1	0	1	1	1	1	0	0	1	0	1	1
1	1	0	1	1	1	1	1	0	0	1	0	1	0
1	0	1	1	1	1	1	1	0	0	1	0	0	1
0	1	1	1	1	1	1	1	0	0	1	0	0	0

The encoder encodes corresponding i/p states to a particular o/p format. The binary encoder encodes the i/p to the corresponding binary format. The encoder o/p is enabled by Enable i/p (EI) signal line at logic '0'. Logic '1' on EI line forces all o/p's to the active high state. The group signal GS is active-low when any i/p is low; this indicates when any i/p is active. The enable o/p (EO) is active-low when all i/p's are at logic 'high'.

74LS148 is a priority encoder and it provides priority encoding of the inputs to ensure that only the highest order data is encoded when multiple data lines are asserted.

c) Summarize analog and mixed signal design in VLSI and Integrated Circuit design (5 M)

Soln.

Analog design - Deals with the design of integrated circuits handling analog signals & data. Analog design gives more emphasis to the physics aspects of semiconductor devices such as gain, power dissipation, resistance etc. RF IC design, Op-amp design, Vg. regulator IC design etc. examples. for analog IC design.

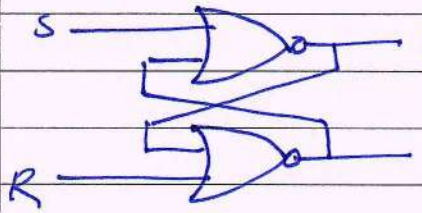
Mixed Signal Design - In today's world most of the applications require the co-existence of digital signals & analog signals for functioning. Mixed signal design involves design of IC's, which handle both digital & analog signals as well as data. Ex - ADC for mixed signal IC.

Q.6) a) Define Flip-Flops. Explain SR FF with truth table & logic circuit. (5 M)

Soln

A flip-flop is a fundamental digital circuit with two stable states (0 & 1) used to store one bit of binary data.

SR Flip Flop -



I/P		O/P	
S	R	Prev. State	Present State
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	X
1	1	1	X

The SR FF is built using two NOR gates. The o/p of each NOR gate is fed back as i/p to the other NOR gate. This ensures that if the o/p of one NOR gate is logic 1, the o/p of other NOR gate will be at logic 0.

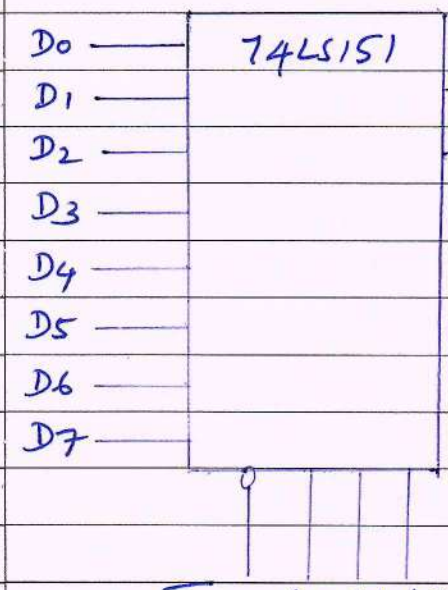
Working of SR FF -

- If the set i/p (S) is at logic 1 & Reset i/p at (R) logic 0, the o/p remains at logic 1, regardless of the previous o/p state.
- If S is at logic 0, & Reset (R) at 1, the o/p remains at logic 0, regardless of the previous o/p.
- If both S & R at logic 0, the o/p remains at previous logic
- If S=R=1, then FF enters in to race around condition & state becomes undefined.

b) Explain 8:1 MUX IC and input/output signal states with Truth Table (5 M).

Soln:

8:1 Multiplexer IC.



Truth Table

Input				output	
A2	A1	A0	$\overline{EN}$	Y	$\overline{Y}$
0	0	0	0	D0	$\overline{D0}$
0	0	1	0	D1	$\overline{D1}$
0	1	0	0	D2	$\overline{D2}$
0	1	1	0	D3	$\overline{D3}$
1	0	0	0	D4	$\overline{D4}$
1	0	1	0	D5	$\overline{D5}$
1	1	0	0	D6	$\overline{D6}$
1	1	1	0	D7	$\overline{D7}$
X	X	X	1	0	1

A multiplexer is a digital s/w which connects one i/p line from a set of i/p lines, to an o/p line at a given pt. of time. It contains multiple i/p lines and a single o/p line. The i/p's of a mux are said to be multiplexed. It is possible to connect one input with the o/p line at a time. The i/p line is selected thro' the MUX control lines.

The mux is enabled only when 'EN' signal line is at logic '0'. A "high" on EN line forces the o/p to the inactive (low state). The i/p line is switched to the o/p line thro' the channel select control lines A2, A1, A0.

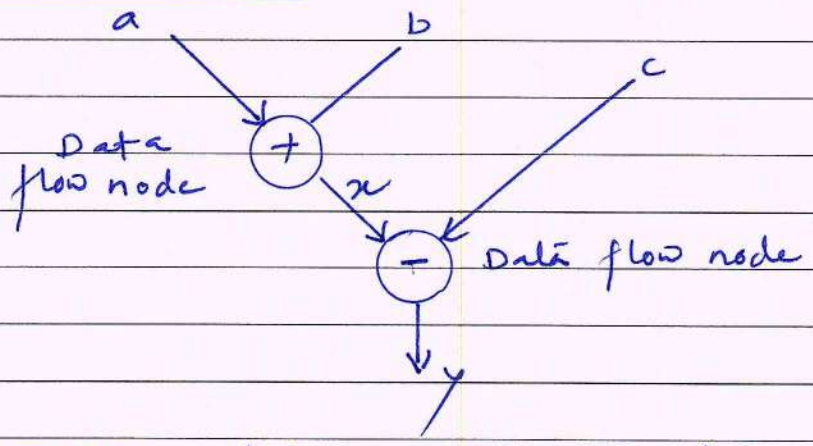
c) List the different computational model in Embedded System. Explain the DFG and CDFG model (10M).

Soln:

The different computational models are;

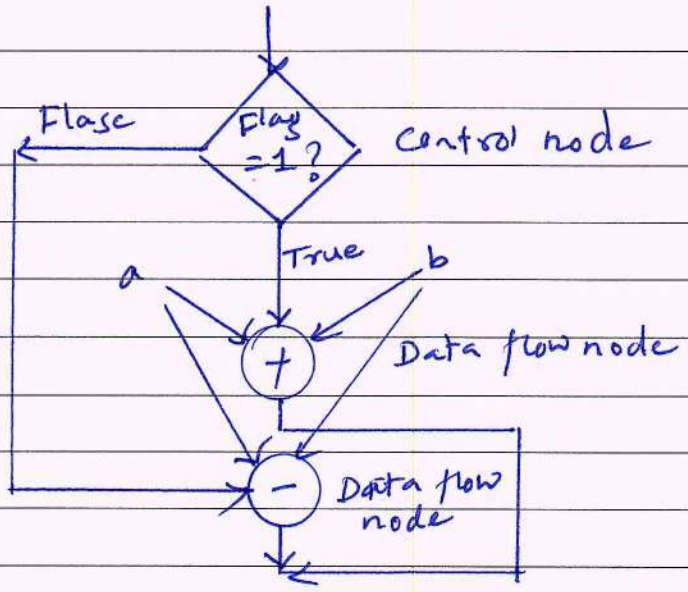
i) Data Flow Graph (DFG) model
ii) State Machine Model
iii) Concurrent Process Model
iv) Sequential Program Model
v) Object Oriented Model

DFG Model -



DFG model translates the data processing requirements in to a data flow graph. DFG is a data driven model in which the program execution is determined by data. This model emphasizes on the data & operations on the data which transforms the i/p data to o/p data. Indeed DFG is a visual model in which the oprn on the data (process) is represented using a blocle (circle) and data flow represented using arrows. An inward arrow to the process represents input data and an outward arrow from the process represents o/p data in DFG notation.

CDFG Model -



CDFG model involves control operations. The CDFG model is used for modelling applications involving conditional program execution. CDFG model contains both data oprns & control oprns. CDFG uses DFG as element and conditional as decision makers. CDFG contains both data flow nodes and decision nodes.

The control node is represented by a "Diamond" blocle which is the decision making element in a normal flow chart based design. CDFG translates the requirement, which is modeled to a concurrent process model. The decision on which process is to be executed is determined by the control node.

Module-4

Q.7) a) What are the two basic approaches used for embedded firmware design. Explain super loop based approach (6 M)

Soln:

The two basic approaches used for embedded firmware design are;

i) Super Loop based approach
ii) Embedded OS based approach.

Super Loop based Approach -

The super loop based firmware development approach is adopted for applications that are not time critical and where the response time is not so important. It is very similar to a conventional procedural programming where the code is executed task by task. The task listed at the top of the program code is executed first and the tasks just below the top are executed after completing the first task. This is a true procedural one. In a multiple task based s/m, each task is executed in serial in this approach.

The firmware execution for this will be;

i) Configure the common parameters & perform initialization for various h/w components memory, registers, etc.
ii) Start the first task & execute it
iii) Execute the second task
iv) Execute the next task
v) :

vi) :
vii) Execute the last defined task
viii) Jump back to the first task and follow the same flow.

The super loop based design does not require an OS, since there is no need for scheduling which task is to be executed and assigning priority to each task. In a super loop based design, the priorities are fixed & the order in which the tasks to be executed are also fixed. Hence the code for performing these tasks will be residing in the code memory without an OS image.

This type of design is deployed in low-cost embedded products & products where response time is not critical.

Ex- Electronic video game toy containing keypad and display unit.

The major drawback of this design is that any failure in any part of a single task may affect the total system and lack of real timeliness.

b) Explain the types of files generated on Cross-compilation (8 M)

Soln.

Various files generated during the cross-compilation or cross-assembling process are;
List file (.lst), Hex file (.hex), Pre-processor output file, Map file (File extension linker dependent), Object file (.obj).

i) List file (.lst) - Listing file is generated during the cross-compilation process and it contains an abundance of information about the cross compilation process, like cross compiler details, formatted source text ('C' code), assembly code generated from the source file, symbol tables, errors & warnings detected during the cross compilation process. This type of information contained in the list file is cross-compiler specific.

ii) Preprocessor Output file - The preprocessor o/p file generated during cross-compilation contains the preprocessor o/p ~~file~~ for the pre-processor instructions used in the source file. Preprocessor o/p file is used for verifying the operation of macros and conditional preprocessor directives. The preprocessor o/p file is a valid 'C' source file. File extension of preprocessor o/p file is cross compiler dependent

iii) Object file (.obj file) - Cross-compiling/assembling each source module (written in C/Assembly) converts the various Embedded C/Assembly instructions and other directives present in the module to an object (.obj) file. The format of the .obj file is cross compiler dependent. OMF51 & OMF52 are the two objects file formats supported by C51 cross compiler. The object file is a specifically formatted file with data records for symbolic information, object code, debugging information,

library references, etc.
The list of some of the details stored in an object file is given below;
1) Reserved memory for global variables
2) Public symbol names
3) External symbol references
4) Library files with which to link
5) Debugging information to help synch source lines with object code.

iv) Map file (.map) — The cross-compiler converts each source code module into a relocatable object (OBJ) file. Cross-compiling each source code module generates its own list file. In a project with multiple list files, the cross compilation of each module generates its own list file. The object files so created are relocatable codes, meaning their location in the code memory is not fixed. It's the responsibility of a linker to link all these object files. The locator is responsible for locating absolute address to each module in the code memory. Linking & locating of relocatable object files will also generate a list file called "linker list file" or "map file".

Map file contains information about link/locate process & is composed of a number of sections. The different sections listed in a map file are compiler dependent. ~~The information generally found by map files is hierarchical.~~

c) Explain the advantages of high level language based development. (6 M).

Soln:

i) Reduced development time - Developer requires less or little knowledge on the internal h/w details and the architecture of the target processor / controller. Base minimal knowledge of the memory organization and register details of the target processor in use and syntax of the HLL are the only pre-requisites of HLL based firmware development. Rest of the things will be taken care of by the cross-compiler used for HLL. Thus the ramp up time required by the developer in understanding the target h/w & target M/c's assembly instructions is waived off by the cross compiler and it reduces the development time by significant reduction in developer effort. HLL based development also refines the scope of embedded firmware development from a team of specialized architects to anyone knowing the syntax of the language and willing to put little effort on understanding the minimal h/w details. With HLL, each task can be accomplished by lesser no of lines of code compared to the target processor / controller specific assembly language based development

ii) Developer independency - The syntax used by most of the HLL's are universal and a program written in HLL can easily be understood by a second person knowing the syntax of the language. Certain instructions may require little knowledge

of the target h/w details like register set, memory map etc. Apart from these, the HLL based firmware development makes the firmware, developer independent. HLL's always instruct certain set of rules for writing the code and commenting the piece of code. If the developer strictly adheres to the rules, the firmware will be 100% developer independent.

iii) Portability - Target applications written in HLL are converted to target processor / controller understandable format (machine codes) by a cross-compiler, with little or less effort by simply re-compiling / little code modification followed by re-compiling the application for the required target processor / controller, provided the cross-compiler has support for the processor / controller selected. This makes applications written in HLL highly portable. Little effort may be required in the existing code to replace the target processor specific header files with new header files, register definitions with new ones etc. This is the major flexibility offered by HLL based design.

Q.8) a) Explain monitor program based firmware debugging with neat sketch (6M)

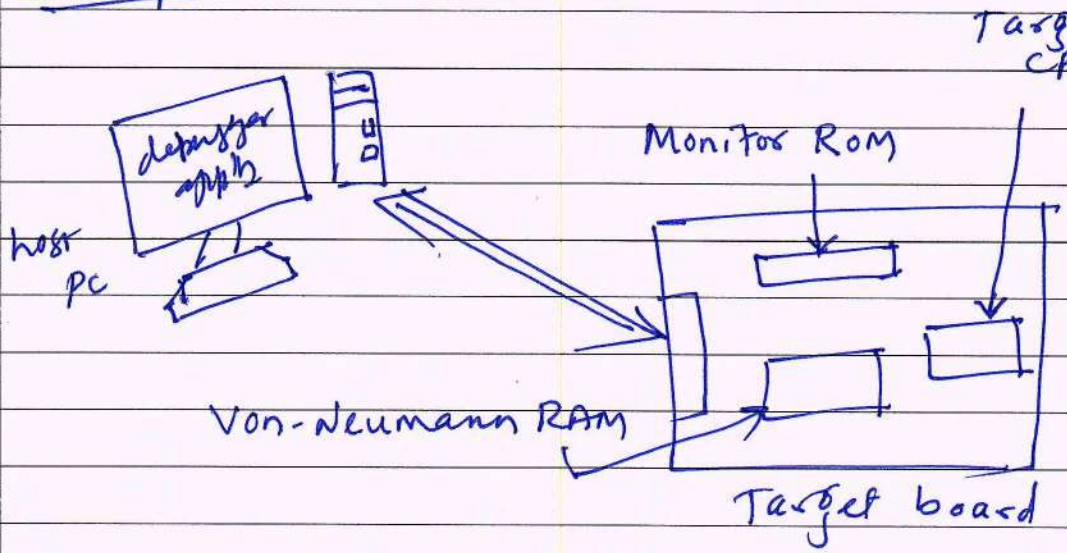
Soln

Monitor program based firmware debugging is the first adopted invasive method for firmware debugging.

In this approach a monitor program which acts as a supervisor is developed. The monitor program controls the downloading of user code in to the

code memory, inspects & modifies register/memory locations; allows single stepping of source code etc. The monitor program implements the debug fns as per a pre defined command set from the debug appln interface. The monitor program always listens to the serial port of the target device & according to the command received from the serial interface it performs command specific actions like firmware downloading, memory/inspection modification, firmware single stepping & sends the debug information back to the main debug program running on the development PC.

Set up.

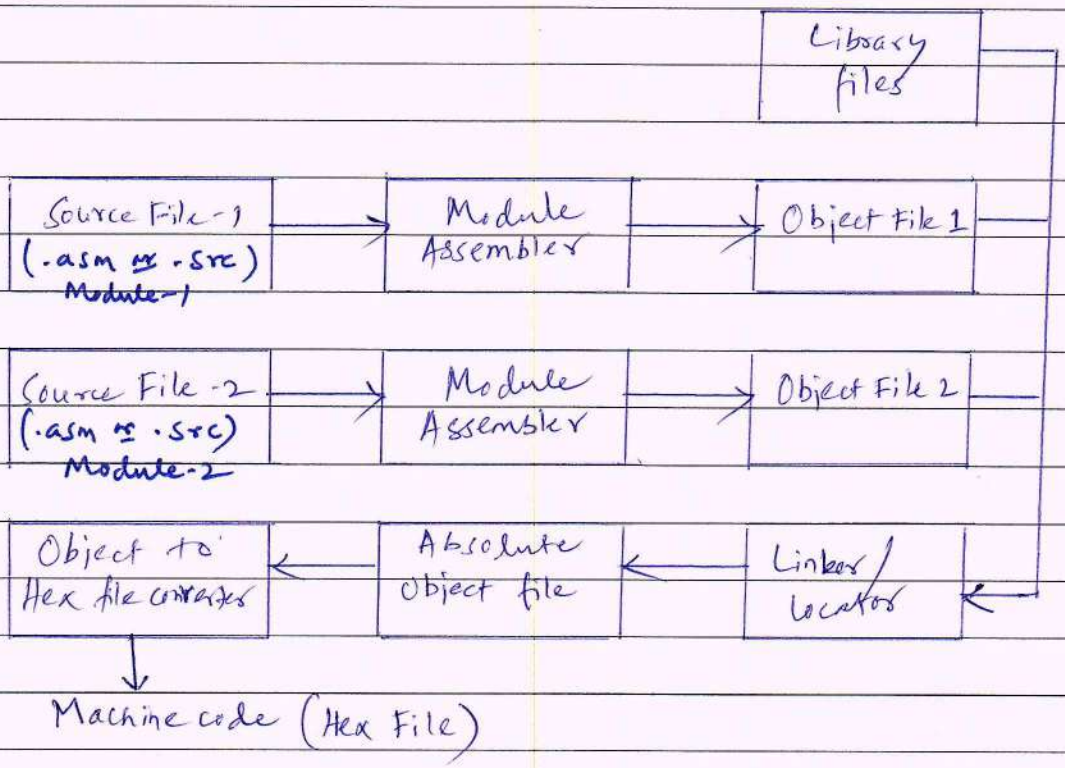


The monitor program contains the following set of minimal features;

1) Command set interface to establish comm with debugging appln
2) Firmware download option to code memory
3) Examine & modify processor registers & RAM
4) Single step program execution
5) Set breakpoints in firmware execution
6) Send debug information to debug appln running on host m/c

b) Explain assembly language to machine language conversion process with neat block diagram (8M).

Soln.



Translation of assembly code to machine code is performed by assemblers. The assemblers for different target M/c's are different and it is common that assemblers from multiple vendors are available in the market for the same target machines.

Each source module is written in assembly & is stored as .src or .asm file. Each file can be assembled separately to examine the syntax errors and incorrect assembly instructions. On successful assembling of each .src / .asm file, a corresponding object file is created with extension ".obj". The object file does not contain the absolute address of where the generated code needs to be placed on the program memory & hence it is called a re-locatable segment.

It can be placed at any code memory location and it is the responsibility of the linker/locator to assign absolute address for this module.

Absolute address location is done at the absolute object file creation stage. Each module can share variables & subroutines (fns) among them. Exploring a variable/function from a module is done by declaring that variable/fn as PUBLIC in the source module.

c) Explain disassembler/decompiler in embedded s/m (6M)

Soln.

Disassembler is a utility program which converts M/c codes in to target processor specific assembly codes/instructions. The process of converting machine codes in to assembly code is known as "disassembling".

In operation, disassembling is complementary to assembling/cross-assembling. Decompiler is the utility program for translating M/c codes into corresponding HLL instructions. Decompiler performs the reverse operation of compiler/cross-compiler. The disassemblers/decompilers for different family of processors/controllers are different. Disassembler/decompilers are deployed in reverse engineering. Disassemblers/decompilers help the reverse engineering process by translating the embedded firmware in to assembly/HLL instructions. Disassemblers/decompilers are powerful tools for analysing the presence of malicious codes in an executable image.

Module-5

Q.9) a) Explain operating system basics in embedded s/m (8M)

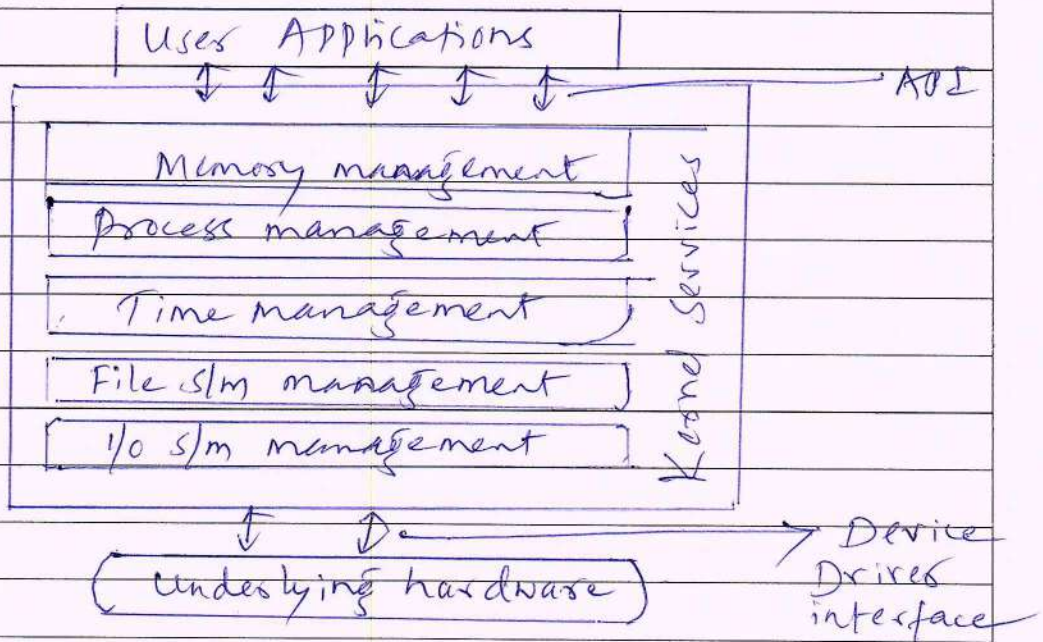
Soln:

Operating System basics -

The OS acts as a bridge b/w the user applications/ tasks and the underlying s/m resources thro' a set of s/m functionalities & services. The OS manages the s/m resources & makes them available to the user applications / tasks on a need basis. The primary fns of OS is;

- Make the s/m convenient to use
- Organise & manage the s/m resources efficie-ntly and correctly.

Basic components of OS -



The Kernel -

Kernel is the core of the OS. It is responsible for managing the s/m resources and the communication

among the h/w & other s/m services. Kernel acts as the abstraction layer b/w s/m resources and user applications. Kernel contains a set of s/m libraries and services. For a general purpose OS, the kernel contains different services for handling;
process management, Primary memory management, File s/m management, I/O s/m management, secondary storage management, protection systems, Interrupt handler.

Process management - It deals with the managing the processes/tasks. It includes setting up the memory space for the process, loading the processes code in to the memory space, allocating s/m resources, scheduling & managing the execution of the processes, setting up & managing the PCB, inter process comm & synchronization, process termination/deletion etc.

Primary memory management - The term primary memory refers to the RAM where processes are loaded and variables and shared data associated with each process are stored. The MMU of the Kernel is responsible for;
i) Keeping track of which part of the memory area is currently used by which process
ii) Allocating & deallocating memory space on a need basis (dynamic memory allocation).

File system Management - File is a collection of information. Each file differ in the kind of information

they hold and the way in which the information is stored. The file operation is a useful service provided by the OS. The file s/m management service of Kernel is responsible for;

- Creation, deletion & alteration of files
- Creation, deletion & alteration of directories
- Saving of files in the secondary storage memory
- providing automatic allocation of file space based on amount of free space available
- Providing flexible naming convention for the files.

I/O system (device) management - Kernel is responsible for routing I/O requests coming from different user applications to the appropriate I/O devices of the s/m. In a well-structured OS, the direct accessing of I/O devices are not allowed and the access to them are provided thro' a set of API's exposed by the Kernel. The Kernel maintains a list of all the I/O devices of the s/m. This list may be available devices as & when a new device is installed. The service "Device Manager" of the Kernel is responsible for handling all I/O devices related oprns. The kernel talks to the I/O device thro' a set of low-level s/m's calls, which are implemented in a service, called device drivers. The device drivers are specific to a device or a class of devices. The device manager is responsible for;

- Loading & Unloading of device drivers
- Exchanging information & the s/m specific control signals to & from the device

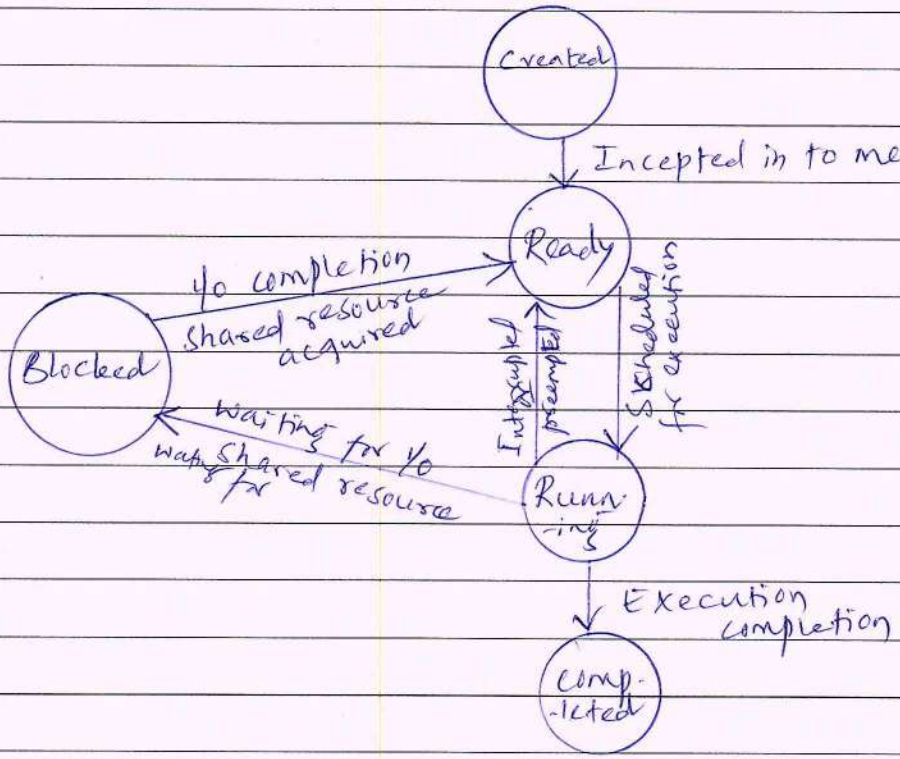
Secondary Storage Management - This deals with the managing the secondary storage memory devices, if any, connected to the s/m. Secondary memory is used as backup medium for programs & data since the main memory is volatile. In most of the systems, the secondary storage is kept in hard disks. The secondary storage management service of Kernel deals with;

- disk storage allocation
- disk scheduling
- free disk space management

Protection systems - Protection deals with implementing the security policies to restrict the access to both user & s/m resources by different applications or processes or users. In multiuser supported OS, one user may not be allowed to view or modify the whole / portions of another's data or profile details. In addition some application may not be granted with permission to make use of some of the s/m resources. This kind of protection is provided by the protection services running within the Kernel.

Interrupt handler - Kernel provides handler mechanism for all external / internal interrupts generated by the s/m.

b) Explain the process states and state transition in embedded s/m with blocle diagram (6 M)



The creation of a process to its termination is not a single step operation. The process traverses thro' a series of states during its transition from the newly created state to the terminated state. The cycle thro' which a process changes its state from "newly created" state to "execution completed" is known as "Process Life Cycle". The various states thro' which a process traverses thro' during a Process Life Cycle indicates the current status of the process. w.r.t. time and also provides information on what it is allowed to do next.

The state at which a process is being created is referred as "Created State". The OS recognises a process in the "created state" but no resources are allocated to the process. The state,

Where the process is incepted into the memory and awaiting the processor time for execution, is known as "Ready state". At this stage, the process is placed in the "Ready List" queue maintained by the OS.

The state where in the source code instructions corresponding to the process is being excented is called "Running state". & in running state proc-cess execution happens. "Blocked state / wait state" refers to a state where a running process is tempor-arily suspended from execution and does not have immediate access to resources. The blocked state may be invoked by various conditions like; the process enters a wait state for an event to occur or waiting for getting access to a shared resource A state where a process completes its execution is known as "completed state". The transition of a process from one state to another is known as "State Transition". When a process changes its state from Ready to running or from running to blocked or terminated or from blocked to running, the CPU allocation for the process may also change.

c) Explain types of multitasking in embedded s/m (6M)

Soln:-

Types of Multitasking -

i) Co-operative multitasking —

It is the most primitive form of multitasking in which a task/process gets a chance to execute only when the currently executing task/process voluntarily relinquishes the CPU. In this process

any task/process can hold the CPU as much time as it wants. Since this type of implementation involves the mercy of the tasks each other for getting the CPU time for execution, it is known as co-operative multitasking. If the currently executing task is non-cooperative, the other tasks may have to wait for a long time to get the CPU.

ii) Preemptive Multitasking -

Preemptive multitasking ensures that every task/process gets a chance to execute. When & how much time a process gets is dependent on the implementation of the preemptive scheduling. As the name indicates, in preemptive multitasking the currently running task/process is preempted to give a chance to other tasks/process to execute. The preemption of task may be based on time slots or tasks/process priority

iii) Non-preemptive multitasking -

Here, the process/task which is currently given the CPU time, is allowed to execute until it terminates or enters the blocked/wait state, waiting for an I/O or system resource. In non-preemptive multitasking the currently executing task relinquishes the CPU when it waits for an I/O or S/W resource or an event to occur.

Q.10) a) Elaborate preemptive scheduling in embedded system (8 M)

Soln.

Preemptive scheduling is employed in systems, which implements preemptive multitasking model. In preemptive scheduling, every task in the "Ready" queue gets a chance to execute. When & how often each process gets a chance to execute (gets the cpu time) is dependent on the type of preemptive scheduling algorithm used for scheduling the process. In this kind of scheduling, the scheduler can preempt (stop temporarily) the currently executing task/process and select another task from the "Ready" queue for execution. When to pre-empt a task and which task is to be picked up from the "Ready" queue for execution after preempting the current task is purely dependent on the scheduling algorithm.

A task which is preempted by the scheduler is moved to the "Ready" queue. The act of moving a "Running" process/task in to the "Ready" queue by the scheduler, without the processes requesting for it is known as "Preemption".

Preemptive scheduling can be implemented in different approaches. The two important approaches adopted in preemptive scheduling are time-based preemption and priority-based preemption.

The various types of preemptive scheduling adopted in task/process scheduling are;

i) Preemptive SJF scheduling/Shortest Remaining Time (SRT)

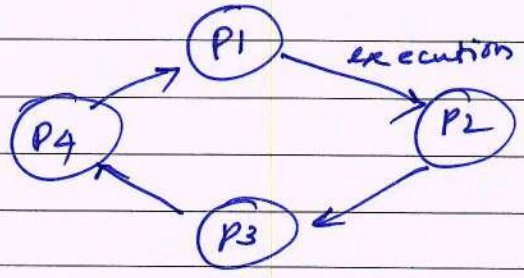
ii) Round Robin (RR) scheduling
iii) Priority based scheduling
iv)

i) Pre-emptive SJF scheduling / SRT -

The non-preemptive SJF scheduling algorithm sorts the "Ready" queue only after completing the execution of the current process or when the process enters "wait" state, where as the preemptive SJF scheduling algorithm sorts the "Ready" queue when a new process enters the "Ready" queue and checks whether the execution time of the new process is shorter than the remaining of the total estimated time for the currently executing process. If the execution of the new process is less, the currently executing process is preempted & the new process is scheduled for execution. Thus the preemptive SJF scheduling always compares the execution completion time of a new process entered the "Ready" queue with the remaining time for completion of the currently executing process and schedules the process with shortest remaining time for execution.

ii) Round Robin (RR) scheduling -

Round Robin scheduling brings the message "Equal Chance to all", and each process in the ready queue is executed for a pre-defined time slot.



The execution starts with picking up the first process in the "Ready" queue. It is executed for a pre-defined time & when the pre-defined time elapses or the process completes, the next process in the "Ready" queue is executed. This is repeated for all the processes in the "Ready" queue Once the each process in the "Ready" queue is executed for the pre-defined time period, the scheduler comes back & picks the first process in the "Ready" queue again for execution. The sequence is repeated.

iii) Priority based scheduling -

It is same as that of the non-preemptive priority based scheduling except for the switching of execution between tasks. In preemptive scheduling, any high priority process entering the "Ready" queue is immediately scheduled for execution whereas in the non-preemptive scheduling any high priority process entering the "Ready" queue is scheduled only after the currently executing the process completes its execution or only when it voluntarily relinquishes the CPU.

b) Explain the types of operating systems in embedded system (6 M).

Soln.

Operating systems in embedded system are classified into two types;

i) General Purpose Operating system (GPOS)
ii) Real-Time operating System (RTOS)

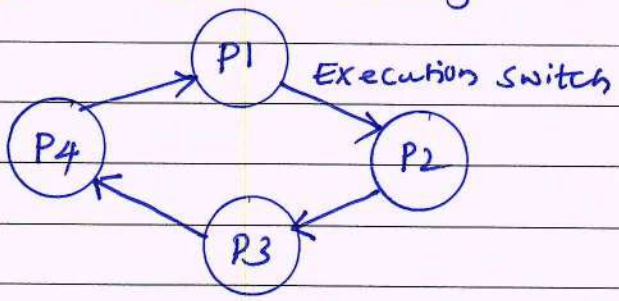
i) General Purpose Operating System (GPOS) -
The operating systems, which are deployed in general computing systems, are referred as "General Purpose Operating s/m's". The Kernel of such an OS is more generalized and it contains all kinds of services required for executing generic applications. GPOS are often quite non-deterministic in behaviour. Their services can inject random delays in to application s/w and may cause low responsiveness of an application at unexpected times. GPOS are usally deployed in computing systems where deterministic behaviour is not an important criterion.
Ex- Personal computer / desktop.

ii) Real-Time Operating System (RTOS) -
In broad sense, 'Real-time' implies deterministic timing behaviour. Deterministic timing behaviour in RTOS context means the OS services consumes only known & expected amounts of time regardless the number of services. RTOS implements policies & rules concerning time-critical allocation of a system resources. The RTOS decides which applications should run in which order and how much time needs to be allocated for each application. Predictable performance is the hallmarke of a well designed RTOS. Rules implement those policies & rules. Policies guide the design of an RTOS.
Ex- VxWorks MicroC/OS-II, Windows Embedded compact, QNX.

c) Explain Round Robin scheduling with block diagram (6M).

Soln.

Round Robin (RR) Scheduling -



Round Robin scheduling is a preemptive CPU scheduling algorithm that assigns each task an equal, fixed time slice, known as a time quantum, in a circular queue, promoting fairness in multitasking systems. When process's time expires, it is preempted and moved to the back of the queue, while the next process runs.

In RR scheduling, each process in the "Ready" queue is executed for a pre-defined timeslot. The execution starts with picking up the first process in the "Ready" queue. It is executed for a pre-defined time & when the time elapses or the process completes (before the pre-defined time slice), the next process in the "Ready" queue is selected for execution. This is repeated for all the processes in the "Ready" queue. Once each process in the "Ready" queue is executed for the pre-defined time period, the scheduler comes back & picks the first process in the "Ready" queue for execution. The sequence is repeated.

Prof 20/02/26.